**Razvoj softvera za embeded sisteme**

**TEMA PROJEKTA:**

|  |
| --- |
| Realizacija bežičnog upravljanja malog auta kontrolisanog od strane android aplikacije |

**TEKST ZADATKA:**

|  |
| --- |
| - Napraviti hardver neophodan za upravljanje auta  - Napisati aplikaciju kojom se programira ESP8266 modul neophodan za bežičnu komunikaciju  - Napisati aplikaciju kojom se programira atmega328p kontroler  - Napisati android aplikaciju |

Mentor Student

Dr Predrag Teodorović Nikola Kovačević e1 6/2018

**Uvod**

Tema ovog projekta je bežično upravljanje malog auta pomoću android aplikacije. Kako bi to moglo da se realizuje potrebno je sledeće:

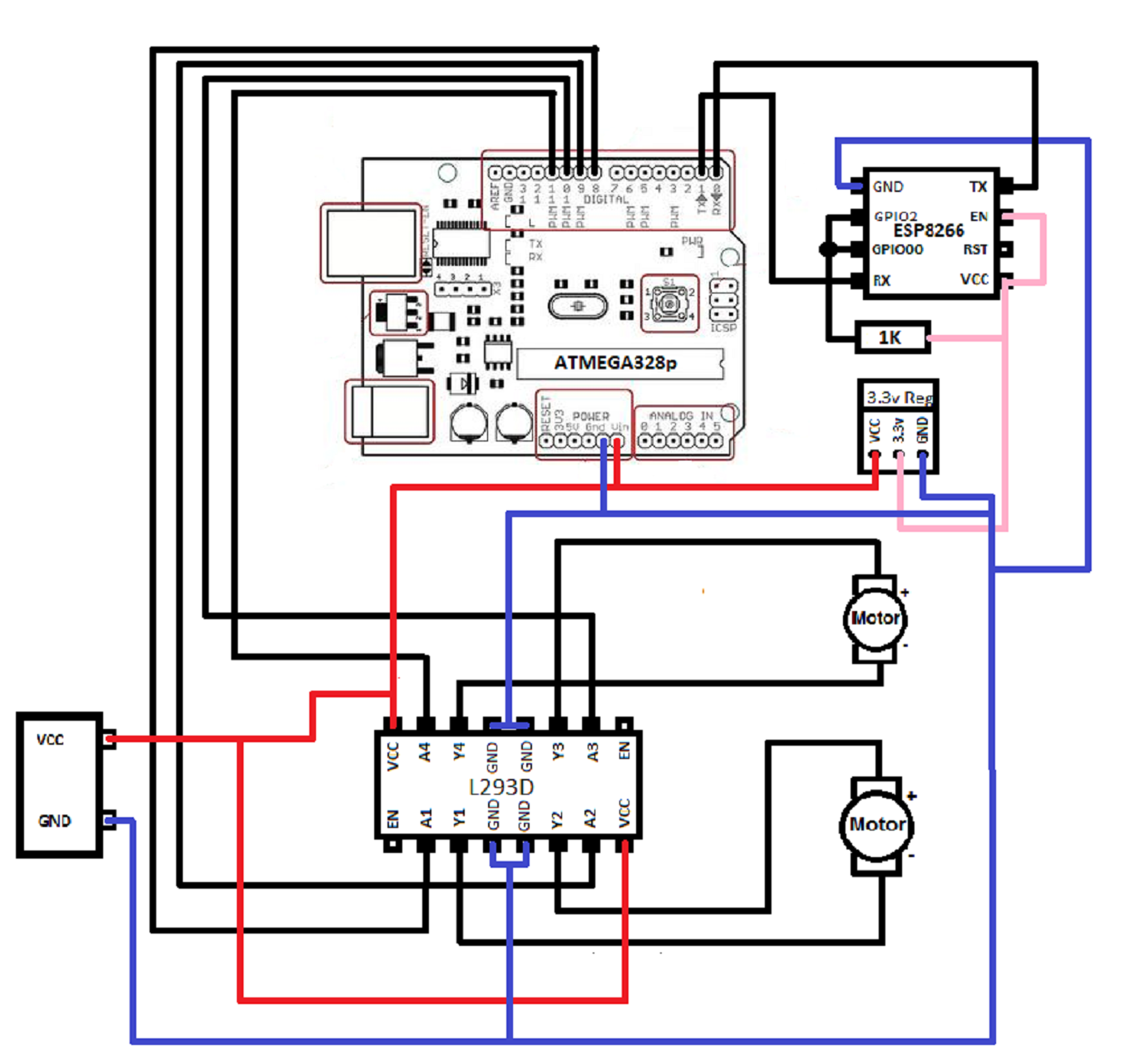
1. Hardver potreban za pokretanje motora auta.
2. Aplikacija koja programira WIFI modul neophodan za bežičnu komunikaciju
3. Aplikacija koja programira atmega328p kontroler koji upravlja hardverom koji pokreće motore
4. Android aplikacija pomoću koje korisnik upravlja autom.

**1. Opis hardvera**

Kako bi se auto pokretao neophodan je sledeći hardver:

1. Atmega328p kontroler u Arduino okruženju.
2. EPS8266 modul za bežičnu komunikaciju.
3. Naponski regulator.
4. 2 DC motora.
5. L293D drajver.
6. Napajanje.

Šematik koji prikazuje način na koji je prethodno navedene komponente potrebno povezati, prikazan je na sledećoj slici:



*Slika 1.*

**DC motor ,L293D komponenta, napajanje**

2 DC motora su neophodna da bi se auto kretao. Jedan motor je zadužen za pokretanje zadnjih točkova, a drugi za pomeranje prednjih točkova u levu ili u desnu stranu kako bi auto mogao da skreće.

Uloga L293D komponente je da na ispravan način pokreće motore. Kao što se sa slike 1 može videti gornji motor je povezan na pinove Y3 i Y4, a donji na pinove Y1 i Y2 i zavisnosti od toga koji od pinova je na visokom, odnosno na niskom nivou motor će se okretati u jednom ili u drugom smeru. Ono što određuje stanje na pinovima Y1, Y2, Y3, Y4 jeste stanje na pinovima A1, A2, A3, A4. Ukoliko se na A1 dovede visok, a na A2 nizak naponski nivo, motor će se okretati u jednom smeru, dok će obrnuto stanje na pinovima A1 i A2 okretati motor u drugom smeru. Isti princip važi i za pinove A3 i A4.

Napajanje koje se koristi jesu 4 redno vezane baterije koje zajedno daju napon od 6v. Razlog za korišćenje ovog napajanje jesu DC motori koji zahtevaju napon od 6v.

**ESP8266, atmega328p, naponski regulator.**

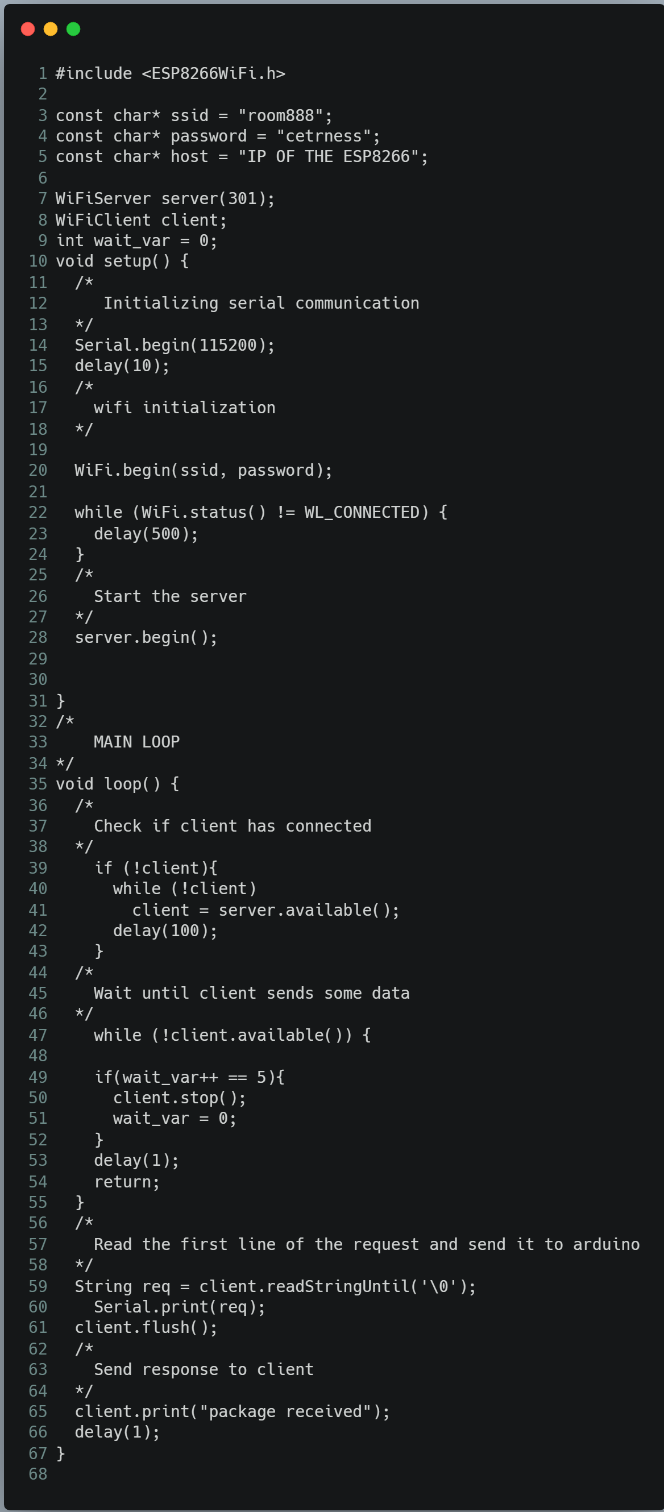
Uloga **ESP8266** modula jeste da obezbedi bežičnu komunikaciju između android aplikacije koju korisnik koristi da bi pokretao automobil i atmega328p kontrolera koji na osnovu pristiglih podataka šalje kontrolne signale kojima upravlja radom motora. Način na koji je to realizovano jeste da android aplikacija putem TCP protokola pošalje podatak ESP8266 modulu čija je uloga da putem serijske komunikacije informaciju prosledi atmega328p kontroleru. Da bi ESP8266 modul ispravno funkcionisao neophodno je uraditi sledeće:

* Na pin za napajanje je potrebno dovesti stabilan napon od 3.3V, i za to je zadužen naposki regulator koji napon sa napajanja od 6V spušta na stabilnih 3.3V.
* Na EN pin je potrebno dovesti visok naponski nivo (3.3V)
* Na Pinove GPIO00 I GPIO02 je preko otpornika od 1kΩ neophodno dovesti visok naponski nivo kako bi se modul konfigurisao da svaki put kada se dovede napajanje on pokrene program koji se nalazi u njegovoj memoriji.
* TX I RX pinove je neophodno respektivno povezati na RX i TX pinove atmega328p kontrolera. Preko TX I RX pinova se obavlja serijska komunikacija sa atmega328p kontrolerom, odnosno preko njih se prosleđuju podaci koje android aplikacija šalje.
* RST pin je ostavljen da visi.

**Atmega328p** je kao što je već prethodno pomenuto zadužena da šalje kontrolne signale ka L293D komponenti, i da njima upravlja radom motora. Način na koji se to postiže jeste preko pinova 8, 9, 10 i 11 arduino okruženja (pogledati sliku 1). Pinovi 8 i 9 su povezani na pinove A1 i A2 L293D komponente respektivno, i postavljanjem visoke vrednosti na pin 8, odnosno niske vrednosti na pin 9 pokrenuće se motor tako da auto ide napred, dok obrnuto stanje na pinovima 8 i 9 dovodi do toga da auto ide nazad. Pinovi 10 i 11 se koriste za upravljanje drugim motorom, odnosno motorom za pokretanje prednjih točkova. Ta dva pina su povezana na A3 i A4 pinove L293D komponente respektivno, i ukoliko je pin 10 na visokom nivou, a pin 11 na niskom prednji točkovi će skrenuti u levo, dok obrnuta situacija skreće točkove u desno. Kako će se generisati kontrolni signali zavisi od informacija koje atmega328p kontroler dobija od strane android aplikacije. Kanal preko koga kontroler prima odnosno šalje informacije je serijski, preko pinova 0 i 1 arduino okruženja i ti pinovi su respektivno povezani na TX i RX pinove ESP8266 komponente.

**2. Opis aplikacije ESP8266 modula**

Kod kojim je opisana aplikacija ESP8266 modula napisan je u Arduino IDE okruženju. I sam kod je prikazan na sledećoj slici:



*Slika 2.*

Kao što se sa slike može videti, kod je jednostavan i u velikoj meri se oslanja na ESP8266WIFI.h biblioteku u kojoj se nalaze klase i metode koje olakšavaju WIFI komunikaciju. Sam kod je razdvojen na dva dela: *Setup* i *loop.*

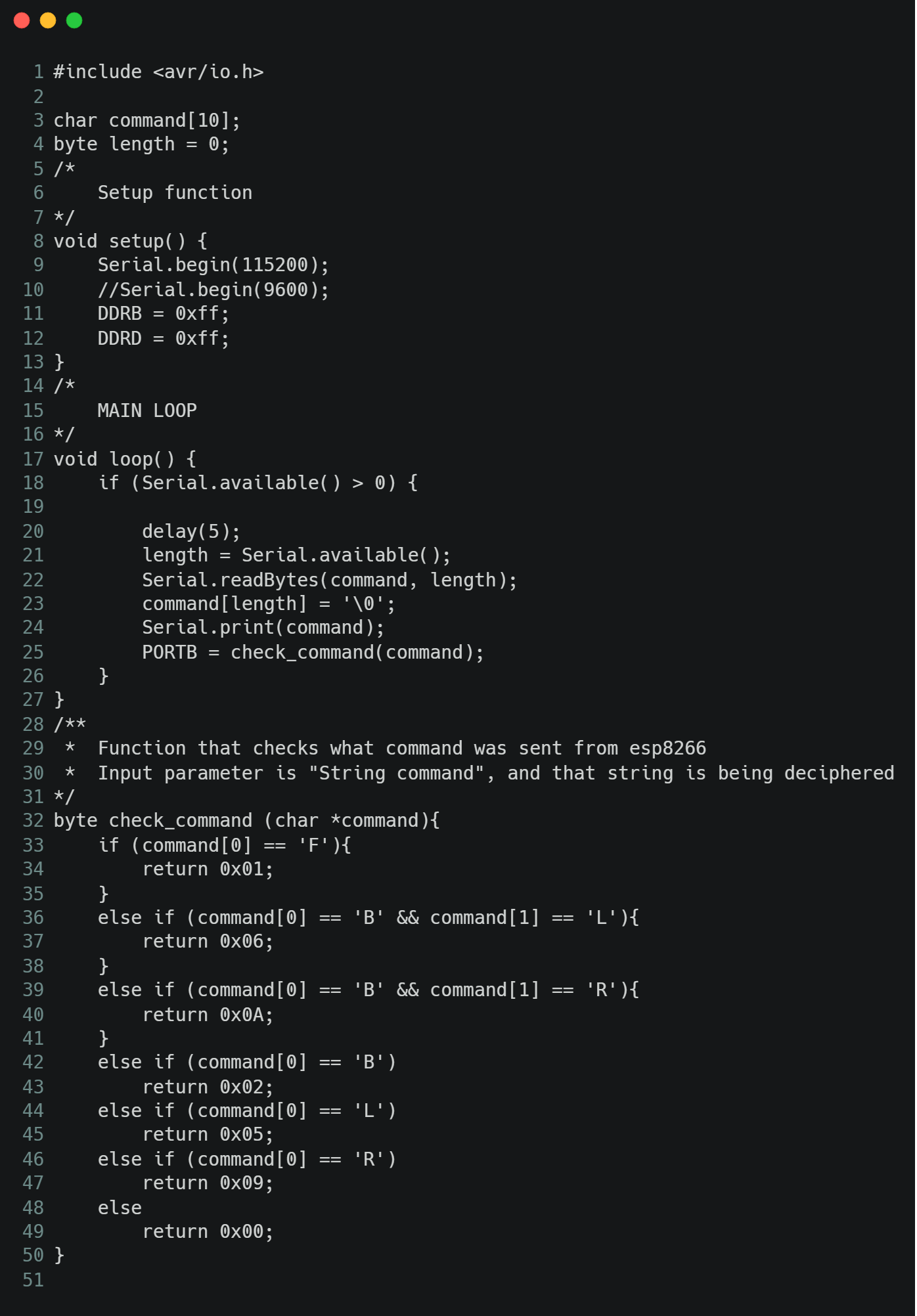
U *setup* delu se vrše osnovna podešavanja modula, odnosno mogućnost pristupa internetu i mogućnost serijske komunikacije. Da bi modul imao pristup internetu neophodno je da se nalazi okolini mreže koja ima WIFI pristup, i čiji su SSID(korisničko ime) i password(šifra) „room888“ i „cetrness“ respektivno. Povezivanje na tu mrežu u prethodnom kodu se obavlja pozivanjem metode *wifi.begin(SSID, password)*, i ukoliko ta mreža postoji modul će se na nju povezati, a ukoliko je nema u blizini modul će u jednoj beskonačnoj pelji čekati da se taj pristup pojavi. Serijska komunikacija je neophodna kako bi ESP8266 mogao da komunicira sa atmega328p kontrolerom, i ona se inicijalizuje pozivanjem metode *Serial.begin(115200)*. Komunikacija između android aplikacije i ESP8266 modula obavlja se preko TCP protokola, pri čemu se ESP8266 ponaša kao server, a android aplikacija kao klijent. Da bi se to realizovalo neophodno je napraviti objekat klase *WifiServer* koja je deklarisana u biblioteci *ESP8266WIFI.h*, kao što je to urađeno u prethodnom kodnom listingu u 7. liniji. Tu je objektu *server* prosleđen broj 301 koji predstavlja port preko koga klijent pristupa serveru. Server se inicijalizuje u 28 liniji koda, i nakon toga klijent može da mu pristupi.

U *loop* delu, se vrše 4 stvari:

* Proverava se da li se neki klijent povezao na server. Ukoliko nije, u petlji se čeka da se neki klijent poveže (40. linija koda, slika 2).
* Nakon što se klijent povezao, u narednoj petlji (47. linija koda, Slika 2) se čeka da klijent pošalje neki podatak, i ako se to ne desi neko vreme, prekida se komunikacija sa tim klijentom i vraća se na prvi korak gde se čeka naredni klijent.
* Ukoliko je klijent poslao informaciju (običan string), ta informacija biva pročitana (59. linija koda, slika 2) i putem serijske komunikacije poslata atmega328p kontroleru.
* Nakon toga server vraća potvrdu da je primio paket (*Package received*), i prethodni koraci se ponavljaju.

**3. Opis aplikacije atmega328p kontrolera**

Kod kojim je programiran atmega328p kontroler je takođe napisan u Arduino IDE okruženju, i prikazan je na sledećoj slici:



*Slika 3*

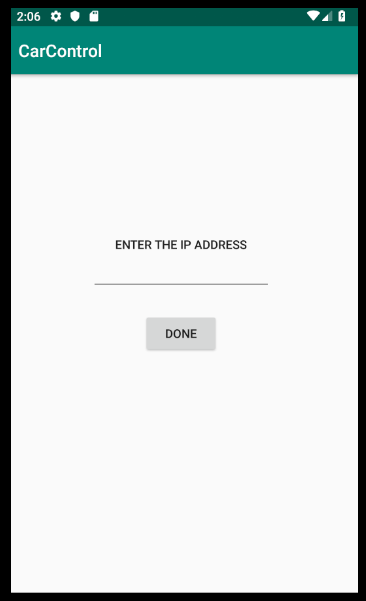
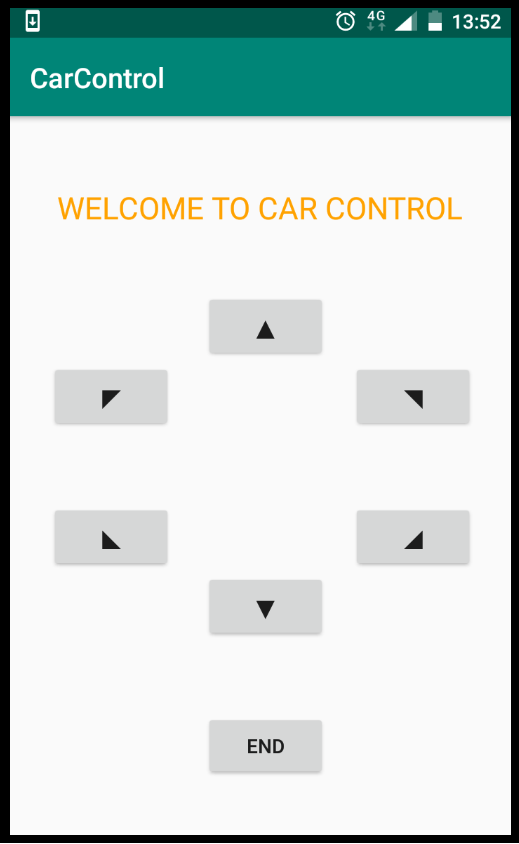
Kod je sačinjen iz dva dela: *setup* i *loop.* U *setup* delu se inicijalizuje UART modul za serijsku komunikaciju pozivom metode *Serial.begin()* i PORTB kontrolera se postavlja da bude izlazni (linija 11, slika 3), jer će se preko njega slati kontrolni signali kojim se upravlja automobilom.

U loop delu se čeka da neka informaciju bude poslata od strane ESP8266 modula. Ono što što ESP8266 modul šalje jeste string na osnovu koga se zaključuje koji kontrolni signali treba da se pošalju preko porta B. Kada se pročita string koji je primljen, on se prosleđuje funkciji koja se zove *check\_command(char \*command)*, i koja je zadužena da na osnovu stringa koji je prosleđen vrati vrednost koja će biti prosleđena na port B i čime će se poslati odgovarajući kontrolni signali. Kod Funkcije se nalazi na prethodnoj slici (linije 32-50). Stringovi koji su od važnosti i na osnovu kojih se generišu kontrolni signali su:

* ‘F’ – ukoliko se primi ovaj string, generišu se signali koji pokreću zadnje točkove, i auto se kreće napred. Ovaj string će na pinove 8, 9, 10, 11 arduino okruženja (slika 1) da postavi vrednosti 1, 0, 0, 0 respektivno. Ukoliko funkcija *check\_command* prepozna ovaj string vratiće vrednost 0x01, i ta vrednost će biti postavljena na port B čime će se na prethodno pomenute pinove postaviti odgovarajuće vrednosti.
* ‘B’ – ukoliko se primi ovaj string, generišu se signali koji pokreću zadnje točkove, i auto se kreće unazad. Ovaj string će na pinove 8, 9, 10, 11 arduino okruženja (slika 1) da postavi vrednosti 0, 1, 0, 0 respektivno.
* ‘L’ – ukoliko se primi ovaj string, generišu se signali koji pokreću prednje točkove i zakreću ih u levo stim da se istovremeno pokreću i zadnji točkovi, tako da će auto ići napred i skretati u levo. Ovaj string će na pinove 8, 9, 10, 11 arduino okruženja (slika 1) da postavi vrednosti 1, 0, 1, 0 respektivno.
* ‘R’ – ukoliko se primi ovaj string, generišu se signali koji pokreću prednje i zadnje točkove, stim što će sada auto skretati u desno. Ovaj string će na pinove 8, 9, 10, 11 arduino okruženja (slika 1) da postavi vrednosti 1, 0, 0, 1 respektivno.
* ‘BL’ – ukoliko se primi ovaj string, generišu se signali koji pokreću prednje točkove i zakreću ih u levo stim da se istovremeno pokreću i zadnji točkovi, tako da auto ide unazad i skreće u levo. Ovaj string će na pinove 8, 9, 10, 11 arduino okruženja (slika 1) da postavi vrednosti 0, 1, 1, 0 respektivno.
* ‘BR’ – ukoliko se primi ovaj string, generišu se signali koji pokreću prednje i zadnje točkove, stim što će sada auto skretati u desno i ići unazad. Ovaj string će na pinove 8, 9, 10, 11 arduino okruženja (slika 1) da postavi vrednosti 0, 1, 0, 1 respektivno.
* Ukoliko se dobije bilo koji drugi string, auto će da stoji. Odnosno na pinovima 8, 9 , 10, 11 arduino okruženja će biti postavljene vrednosti 0, 0, 0, 0 respektivno.

**4. Opis android aplikacije**

Android aplikacija omogućava korisniku da preko svog mobilnog telefona(sa android OS-om) upravlja malim automobilom. Korisnički interfejst se može videte na sledećim slikama:

*Slika 4*  *Slika 5.*

Kada se prvi put uključi android aplikacija, otvara se prozor prikazan na slici 4, i tu se korisniku omogućava da unese IP adresu na kojoj se nalazi ESP8266 modul. Kada se unese adresa i pritisne dugme *DONE*, otvara se prozor prikazan na desnoj strani, u kome korisnik ima pristup kontrolama kojima ukoliko je uneta ispravna IP adresa može da upravlja automobilom. Način korišćenja kontrola je jednostavan, dokle god korisnik drži prst na nekom od dugmića, auto će u tom pravcu ići, dok kada pusti dugme, auto staje. *END* dugme vraća korisnika na početni ekran (slika 4).

**4.1 Opis kod android aplikacije**

Kod neophodan da bi aplikacija funkcionisala sastoji se od 3 klase:

* MainActivity
* Controller
* TCP\_comm

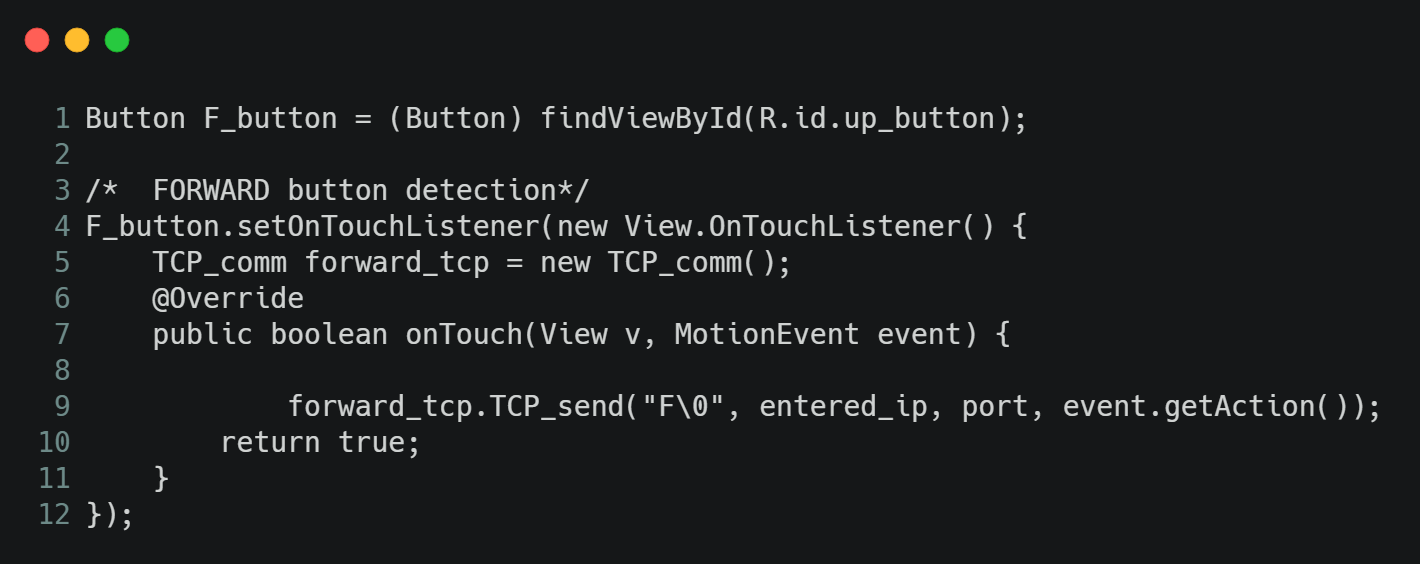
**MainActivity je** glavna klasa i u njoj je implementirana funkcionalnost prozora koji se pokreće pri otvaranju aplikacije (slika 4). Kod koji je implementira prikazan je na sledećoj slici:



*Slika 6*

U ovoj klasi najbitnija je *ClickMe* metoda, čija je uloga da kada se unese IP adresa promeni trenutni prozor na prozor sa kontrolama, i da njemu prosledi string koji predstavlja IP adresu koja je uneta.

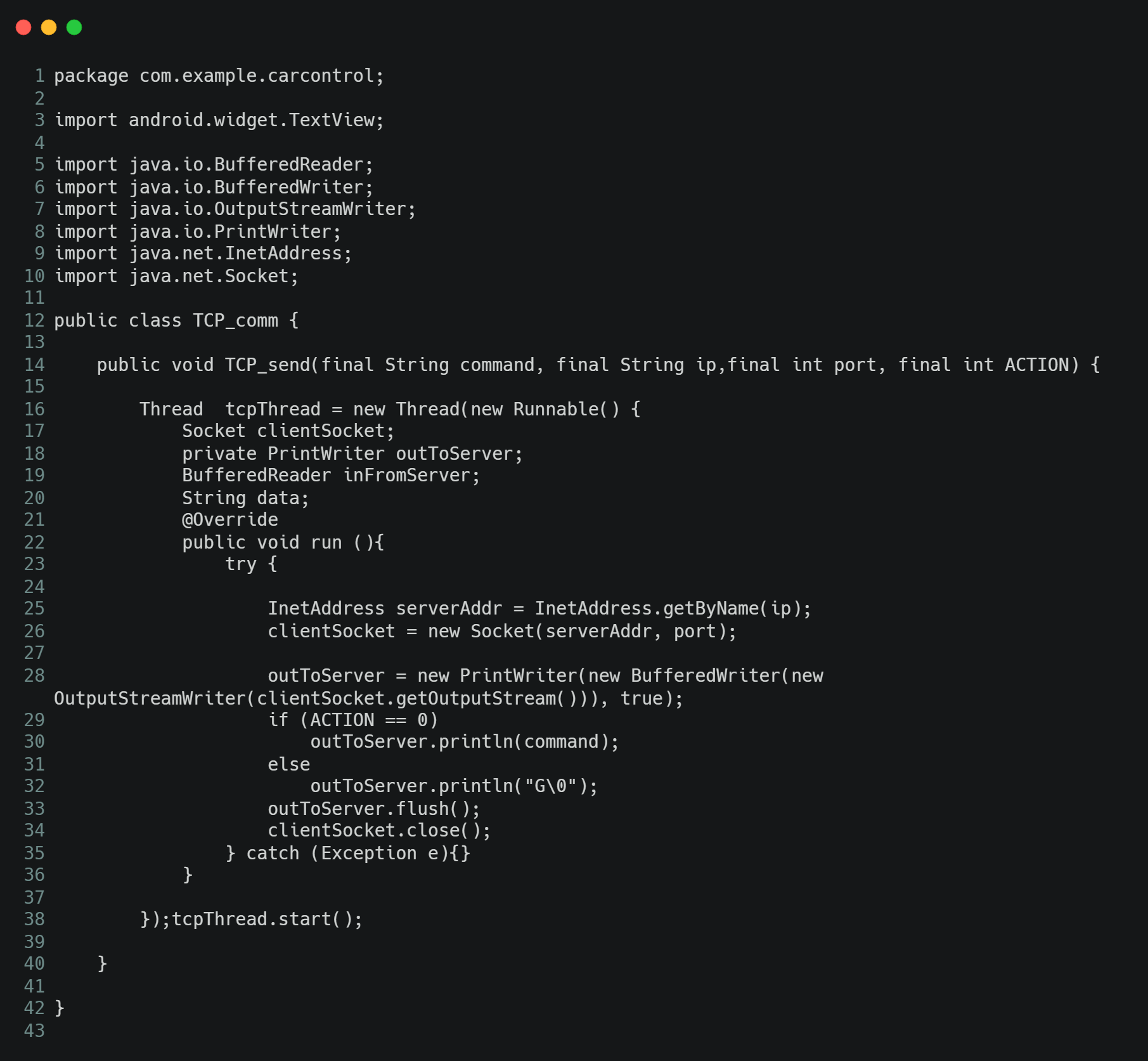
**Controller** klasa implementira funkcionalnost prozora sa slike 5, odnosno u zavisnosti od toga koje je dugme na ekranu pritisnuto, pomoću TCP komunikacije će se ESP8266 modulu proslediti određeni string. Unutar ove klase su implementirane metode za svako dugme na ekranu. Funkcionalnost kontrolnih dugmića je realizovana tako da kada se pritisne ili pusti dugme odgovarajuća metoda biva pozvana. Kako je realizacija metoda za svako dugme praktično ista, na sledećoj slici biće prikazana implementacije samo jedne od njih:

nu

*Slika 7*

Kao što se sa slike 7 može videti kao parametar metode *SetOnTouchListener* se prosleđuje objekat klase *View.OnTouchListener,* pri čemu se istovremeno ta klasa opisuje. Unutar same klase se implementira metoda onTouch, koja je se poziva svaki put kada se dugme pritisne ili pusti. Takođe se unutar te klase pravi i objekat klase *TCP\_comm,* koja će detaljo biti objašnjena kasnije. Za sada je dovoljno da se zna da ona implementira unutar sebe metodu *TCP\_send* pomoću koje se obavlja TCP komunikacija.Metoda je realizovana tako da je neophodno da joj se prosledi string koji predstavja komandu, string koji predstavlja IP, broj porta i informacija da li je dugme pritisnuto ili pušteno(event.getAction()). Kada se ova metoda izvrši , putem TCP protokola informacija je poslata ESP8266 modulu. Za ostale modetode koje implementiraju funkcionalnost ostalih dumgića pogledati izvorni kod.

**TCP\_comm** klasa unutar sebe implementira metodu *TCP\_send,* i njen kod je prikazan na sledećoj slici:



*Slika 8*

Metoda je realizovana kao nit (*eng. Thread*). Da bi se to implementiralo, neophodno je napraviti objetak tipa Thread, i prilikom pozivanja konstruktora kao parametar je potrebno proslediti objekat klase Runnable (linija 16, slika 8). Unutar klase Runnable, neohodno je implementirati metodu *run()* koja će se izvršavati kao zasebna nit. Unutar te metode je implementirana TCP komunikacija, odnosno pravi se novi *clientSocket (linija 26)* i povezuje se na server na datoj IP adresi i datom portu. U ostatku koda se u zavisnosti od parametra ACTION koji se prosleđuje metodi *TCP\_send* i koji daje znak da li je dugme pritisnuto ili pušteno šalje string sa komandom(da se autić pokrene), ili string „G\0“ koji šalje komandu da autić stane.

**Napomena**

Izvorni kodovi mogu se naći na sledećem linku:

<https://github.com/Nikola2444/CarControl_esp8266>